Table of Contents

PHP 开发者实践

PHP Developer Prepares

坚持了 5 年的创业项目决定结束了（行业垂直搜索方向，理想很美好、现实很骨感），作为没混好的草根站长 & 没被风吹起来的小互联网公司技术合伙人，个人基本见证了这几年 PHP 环境的发展（当然主要归功于互联网创业风潮大爆发），感受了一些 PHP 团队和从业者的现状；同时，我们自身在 PHP 研发团队组建过程中也遇到了不少问题，一直想把创业过程中遇到的这些团队实践相关问题整理、总结一下，目前只是个初始版本，欢迎大家拍砖 & 加入！

技术的发展日新月异，我会持续维护、跟进这个项目，欢迎各位有兴趣的朋友 提交建议、问题 - Issue 或 参与贡献、分享 - Pull Request。

- 本项目排版遵循 文案排版指北 和 Stack Overflow Markdown 规范
- 本项目使用了 GitBook 并已发布在 GitBook.com 上，关于如何使用 Gitbook 编写、生成、发布一本在线图书，请移步 GitBook Documentation 或 Gitbook 使用入门 - 中文

> - GitBook.com 是一个很优秀的社区，上面有很多优秀作者自出版自己的著作，就好像 Leanpub
> - Gitbook 是 GitBook.com 提供的一个开源的基于 Node.js 开发的配套工具 - Github 地址

- 开发过程中遇到的绝大多数问题实际上都可以通过搜索引擎找到，关于搜索引擎使用技巧，请参考 如何用好 Google 等搜索引擎？
- 开发过程中遇到问题在论坛、社区中提问是很常见的情况，如何让他人快速理解你的问题、同时自己如何在提问中学习成长，推荐阅读： How To Ask Questions The Smart Way，提问的智慧 - 中文版 或 提问的智慧 - 图片版

> 「《提问的智慧》就是一个敲门砖，它把黑客间的礼仪和准则明白地写下来。它会让你了解到一个事实，为什么那些看起来很牛的人几乎从不提问，似乎他们一进入这个行业就是牛人了。不是的，他们也有问题，但是通常在提问之前就自己解决了；不是因为他们本来就懂得怎么解决，而是解决问题的经历让他们成为牛人；最终，你只会看到网络上多了一篇文章：关于解决XXX问题的方案。」-- Rei

- 因为关于如何使用 PHP 语言本身相关资料已有很多，本文将尽量不涉及 PHP 语言本身、且优先引用已有资料，主要围绕关心 PHP 项目开发技巧和具体实践，通过相关工具和经验的分享，使大家在项目中更好的使用 PHP 技术。

在线阅读

> 重要说明：正在持续修改之中，大部分章节都没有写完，正式发布还需要一段时间，所有内容随时可能发生变动。

- http://ryancao.gitbooks.io/php-developer-prepares/content/

如何参与

本项目在 Github 上维护，欢迎参与：https://github.com/zacao/php-developer-prepares。

- 在 GitHub 上把本项目 `fork` 到自己的仓库，如 `<your-username>/php-developer-prepares`，然后 `clone` 到本地，并设置用户信息。

```
$ git clone git@github.com:<your-username>/php-developer-prepares.git
$ cd php-developer-prepares
$ git config user.name "yourname"
$ git config user.email "your email"
```

- 修改代码后提交，并推送到自己的仓库。

```
$ #do some change on the content
$ git commit -am "Fix issue #1: change helo to hello"
$ git push
```

- 在 GitHub 网站上提交 pull request。
- 定期使用项目仓库内容更新自己仓库内容。

```
$ git remote add upstream https://github.com/zacao/php-developer-prepares
$ git fetch upstream
$ git checkout master
$ git rebase upstream/master
$ git push -f origin master
```

授权许可

本文采用创意共享「署名-非商业性使用」许可证（Creative Commons Attribution-NonCommercial license）。所有内容不仅可以免费阅读，还可以自由使用（比如转载），只需遵守两个条件：

- 署名：必须保留原作者的署名。
- 非商业性使用：除非得到正式许可，否则不得用于商业目的。

前言

近年来，越来越多的 Web 开发人员投入 Python, Node.js, Ruby 的怀抱，与此同时 PHP 也越来越多被人诟病，尽管 PHP 仍然是目前使用最广泛、重要的 Web 开发语言之一。经过几年观察，我发现造成国内这一奇怪现象的原因很大程度上是由于近几年市场对 PHP 开发人员井喷式需求导致大量未经良好训练的 PHP 新兵涌入、原有 PHP 老人多是站长出身且知识结构又未能及时与近几年 PHP 社区更新同步、大量基于原有实践开发的开源项目影响等因素的多重叠加。不论将来后端语言谁是最后的王者，至少在近段时间内 LA/NMP(Linux, Apache/Nginx, MySQL, PHP) 技术仍然是众多中小互联网创业公司的首选技术栈之一，我们尝试在新项目开发过程中更好的使用 PHP 技术，通过结合国外 PHP 领域最新的开发模式、工具和经验，使您的 PHP 项目、团队焕发新生，重装上阵。

源起

作为一名草根码农，接触 PHP 的时间其实不晚（ASP + Access 虚拟空间满天飞的年代，同等价位标配 MySQL 的 PHP 4.0 空间是众多草根站长的最爱）。

种种原因，重新拿起 PHP 已是多年之后，再次踏入 LA/NMP 阵营真心不是因为 PHP 是世界上最好的语言（无意挑起战争 :P）

DiaoSi 互联网创业起步，

- Java 平台架构的繁杂（出门野营还是瑞士军刀短小精悍，随身一大箱专业扳手谁用谁知道 :-(），
- Python & Ruby 是优雅，但国内社区的小众（小公司招人真心难），
- C#？（Windows & SQL Server 服务器正版授权咱就直接跪了，还真心不是安全问题，当然前提是您有牛x系统管理员），
- Perl（上古时代的程序员哪里找，后续接盘侠估计更是欲哭无泪）？ C/C++（程序还没开发出来估计公司就挂了）？
- Go？ Erlang？Scala。。。（一个是资料少、一个是可能相关支持开发包都还没有或坑太多、再一个您招到人再说。。。）

对比起来，LA/NMP 阵营良好的社区群众基础(程序员多啊)、大量的成功开源项目以及简单直接的高开发效率，作为小型互联网创业公司的我们最后选择了 PHP~

> 几年以后的今天，尽管在国内 Python、Node.js、Ruby 等社区有了很大发展、也有越来越多的公司即将或正在使用这些技术，但目前从各大公司招聘情况看来 PHP 仍然是大量中小互联网公司的首选后端语言之一。

在这几年的团队组建与培养过程中，注意到一些有意思的现象，

- 一方面：PHP 社区愈加强调协作&强化性能优势，如 PSR 标准化、 Composer 包管理、HHVM 的风生水起(PHP 7 也同样值得期待)、 phalcon的不断成熟(向鸟哥的 Yaf 致敬)、性能彪悍的 swoole，
- 另一方面：关于 PHP 的吐槽不断，比如大家都知道的一些梗。。。

也许是因为 PHP 的简单直接、易上手造成草根程序员太多（因为未接受科班教育的原因，许多草根程序员都会有知识结构不系统的问题），而火爆的互联网行情进一步造成了开发群体高低水平层次的两极分化。 关于如何具体学习、使用 PHP 技术进行 Web 开发，市面上现在已经有了太多代码与资料，然而关于 PHP 研发人员（其实很多内容同样适用于 PHP 之外的其他语言）在项目中的具体实践却缺少相关系统性资料，这也是我尝试总结、整理此文的原因之一。

资源

- PHP: The Right Way, 中文版
- PHP Best Practices, 中文版

> 强烈建议您直接过一遍上面内容，这样这里的大部分您都可以快速略过，不过我们按照软件开发流程顺序对内容进行组织，同时会有穿插一些中国特色的问题及处理方法 :-)

科学上网 (Cross the GFW)

把科学上网放在最前面，是因为个人认为 好奇心 **&** 动手能力 是许多优秀开发人员共有的特质；更为重要的是，在墙高度不定期加码的情况下，时刻掌握最新科学上网方法能一定程度上反应使用者对于网络协议的基本理解、服务器的操作管理能力、动手能力以及自我学习能力；同样的，各主流社区交流都以英文为主，计算机领域的大牛、信息多富集于中华大局域网之外，要想掌握最新的领域知识、获得第一手行业资料，熟练使用 **Google** 而非百度、使用英文而非中文 是技术人员的良好素质。

目前出墙的方式主要有，

> 注：如果您想给家庭/公司路由器折腾自动翻墙，那么建议您参考这个项目， 一个适用于OpenWRT的全平台翻墙路由方案

VPN 全局模式

- PPTP 协议
- L2TP/IPsec 协议
- OpenVPN

对比代理模式，VPN 模式是最容易/优先被墙识别并屏蔽的，因此出于稳定性和可维护性的考虑，如果您切实需要使用 VPN 方式，个人推荐使用专业 VPN 服务商，而非自建 VPN：

- 云梯 - 稳定、服务好
- 曲径 - 价格稍贵，但物有所值

代理模式

- **Goagent** - GoAgent 是利用 Google App Engine 的服务器资源，使用 Python 开发的代理软件。
- **Shadowsocks** - 由于 Shadowsocks 使用 Socks 协议和可自定义密码的工业级算法加密，使得流量在网络传输过程中不易被他人读取。但是使用不可靠来源的 Shadowsocks 服务器可能会导致使用者的信息泄露。
- **SSH tunnel** - 当用 SSH 连接到跳板机之后，SSH 可以在本地开启一个端口，本地的应用程序连接到本地的这个端口。相当于在本地建立了一个 Socks 代理服务器为本地的应用程序提供 Socks 代理。而这个 Socks 代理通过跳板机连接外网，Socks 代理和跳板机直接的数据通信是在 SSH 隧道里进行的，是安全的。
  更多关于使用 SSH 进行端口转发的信息请参考 - IBM DeveloperWorks: 实战 SSH 端口转发

> 因为代理模式简单易上手且完全可以很好的满足日常的上网需求，所以推荐您优先尝试代理模式。 如果您没有资源或精力自行搭建、维护翻墙环境，推荐您使用专业的服务商，如 红杏（源自「一枝红杏出墙来」）、鱼摆摆 - 仅 Mac OS 系统、Shadowsocks.com 等

相关工具：

> 更多常用代理工具软件（如 Proxychains, Proxifier）的介绍、比较请参考 - **Wikipedia: Comparison of proxifiers**

- **SSH** 客户端 - （Windows 环境适用，SSH 对于 Linux、Mac 那都不是事儿）。
PuTTY, Bitvise, WinSCP, SecureCRT, Xshell...，具体用哪个就仁者见仁了，更多信息请参考 - Comparison of SSH Clients
- **ProxyChains** - Run any program through HTTP or SOCKS proxy。
Proxychains 可以强制一个应用程序使用代理而对整个系统没有影响，它支持 HTTP、Socks、Socks5 这3种协议。
- **Polipo** - Socks Proxy to HTTP Proxy。 因为很多情况下得到的都是 Socks 协议的代理，可以用 Polipo 很方便的将 Socks 代理转为 HTTP 代理。
- **SwitchyOmega** - Manage and switch between multiple proxies quickly & easily。
Google Chrome 浏览器的代理插件，轻松快捷地管理和切换多个代理设置。用于替代 SwitchySharp 插件的更新版本。
- **FoxyProxy** - Advanced proxy management tool for Firefox。
FoxyProxy是Firefox下一款优秀的代理服务器管理扩展，它主要功能是管理Firefox代理服务器设置
- **Proxifier**（收费） - Run any program through HTTP or SOCKS proxy for Windows & Mac。
Proxifier 是一款功能非常强大的 Socks5 客户端，可以让不提供代理服务器设置的网络程序使用 HTTPS 或 Socks5 代理。支持 64 位系统，支持 XP、Vista、Win7、MAC OS，支持 Socks4，Socks5，HTTP 代理协议，兼容性非常好。

LA/NMP 技术介绍

LAMP 是 Web 开发中最流行的模式，即 Linux + Apache + MySQL + PHP。近年已经有一些变化，比如 Web Server 除了 Apache 外，还有 Nginx 和 Lighttpd；数据库除了 MySQL 外，还有各种 NoSQL 引擎；PHP 引擎除了 Zend 之外，还有 HHVM。所以 LAMP 不再指的是具体软件的集合，而是 Linux + Web Server + PHP 语言 + DB 这种开发模式。

下面我们会简要介绍目前最常见的 LA/NMP 模式（Linux + Apache/Nginx + MySQL/MongoDB + PHP）。

Linux 操作系统

对于 PHP 开发人员个人而言，用什么操作系统写代码并不重要，重要的是适合，用着顺手、用着爽。。。

> @使用 Windows 的 PHP 开发同学们，鉴于 Windows 命令行窗口（cmd.exe）一如既往、毫无改进的表现，强烈推荐大家使用 Cmder 替代之。在中文环境下，Cmder 默认安装完毕后您可能会遇到一些问题，解决办法在此 - Cmder 简单使用小结

但是，毕竟PHP 程序多数情况都会运行在 Linux 服务器上，因此个人认为每个 PHP 开发人员都应该熟悉 Linux 系统常用操作以及相关 LA/NMP 环境的搭建、配置。

关于 Linux 发行版

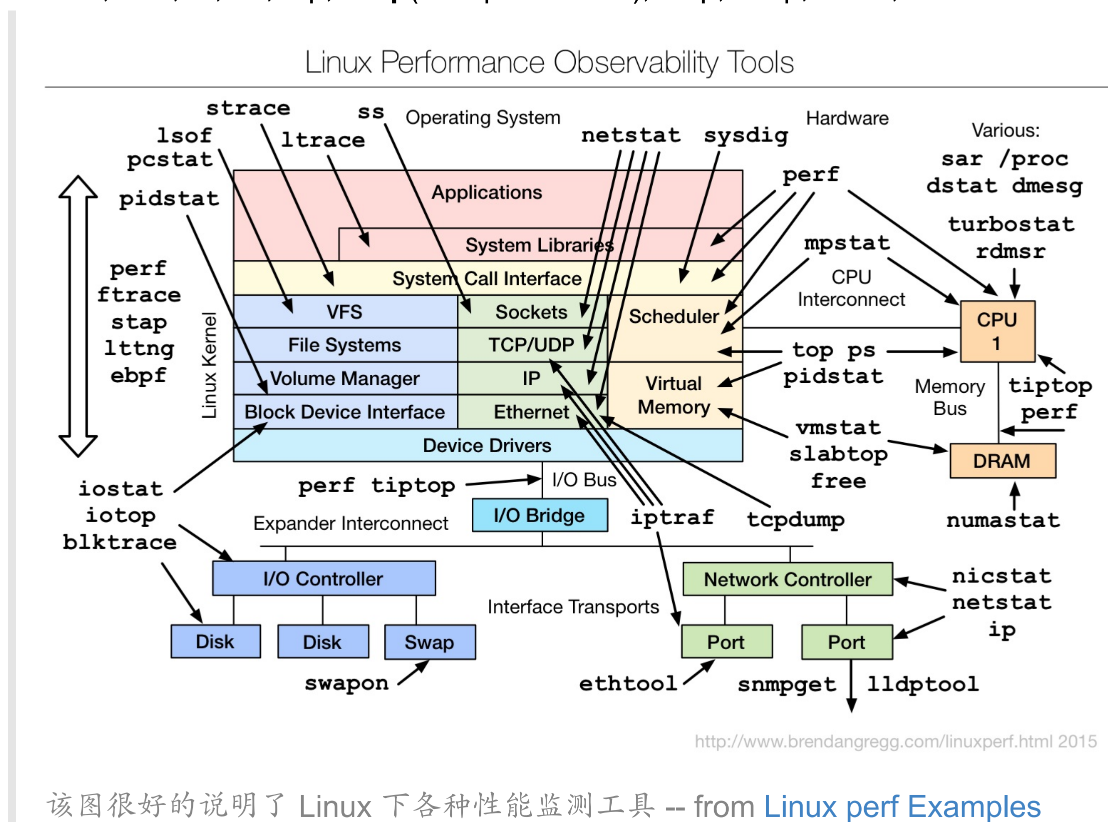
关于 Linux 发行版的选择问题（好吧，又是坑。。。），不过从服务器用户数量来讲，Redhat 系、SUSE 系 与 Ubuntu 系占了多数 。相对比而言，个人更喜欢 Redhat系的 CentOS（关于 Fedora, Redhat 和 CentOS 三者的关系可以 参考这里 ），个人觉得 CentOS 跑服务器相对稳定些（受鸟哥影响较深）、系统本身也干净简单，缺点是很多包官方源里面没有，就算是 EPEL 里面也没多少东西，因此对于 LA/NMP 开发，这里推荐下 EPEL源 + IUS 源，具体的配置方法可以 参考这里 。

一些 Linux 命令/工具

> 粗体部分 - 个人认为应该多加关注的

- Shell
  - echo, printf, **test**（写shell脚本判断条件 -n -z 傻傻分不清楚的时候就靠 ta 了）, sleep, sh, set
- 用户环境
  - clear, env, export, history, passwd, su, sudo, exit, who
- 包管理
  - yum, rpm or apt-get, dpkg
- 文件系统
  - pwd, cd, cp, ls, mkdir, mv, rm, ln
  - file, du, df, mount, touch, cat
  - chmod, chown, chgrp, chksum
- 查找
  - **find**, **grep**, whatis, whereis

- 文本处理
    - **awk**, **sed**, cut, diff, sort, uniq, wc, **xargs**
    - head, **tail**（tail -f）, less
- 进程相关
    - at, cron, **crontab**, kill, killall, pgrep, **ps**, time, nohup, **screen**（远程会话管理）, **supervisord**（进程守护神）
- 网络
    - dig, host, ifconfig, netstat, nslookup, ping, route, traceroute, **iptables**, wget, **curl**
- 压缩
    - **tar**, gzip, unzip
- 远程
    - ssh, scp, sftp, **rsync**（文件同步）
- 系统状况
    - uname, free, df, du, top, **htop**(比 top 好用多了), iftop, iotop, iostat, isof



该图很好的说明了 Linux 下各种性能监测工具 -- from Linux perf Examples

- 其他
    - **man** (查阅命令帮助信息), alias, **ab**（简单性能测试）, service, chkconfig, **ulimit**, **sysctl**

扩展阅读：

- 鸟哥的 **Linux** 私房菜 - 简体版，此鸟哥是台湾的 Linux 大神鸟哥，不是 Laruence 鸟哥 :-)，喜欢原汁原味繁体版的 请看这里 （当年还没出书的时候偶就看这里了，还顺便知道

了一些鸟哥的轶事，比如飞蚊症、服兵役之类的。。。），不喜欢电子版的亦可以 支持纸质版

- Linux工具快速教程 - 这本书专注于Linux工具的最常用用法，以便读者能以最快时间掌握，并在工作中应用；
- 应该知道的 **Linux** 技巧 - Coolshell 陈皓前辈的一些总结，从前辈博客上学到了太多东西。。。 陈皓前辈更多关于 Linux 的文章
- **Redhat** 系转 **Ubuntu** 系快速上手 -英文， Ubuntu 为了抢用户也是拼了 :-)
- 知乎： **Linux** 坑之 **- Linux** 下为何要关闭 **SELinux**？ - 保持 SELinux 默认开启的情况下，你在开发过程中可能会遇到各种诡异问题，除非贵司对安全性有异常高的要求且有专业运维人员，个人还是建议装完系统就直接关闭 SELinux。

Web 应用服务器 - Apache, Nginx

近几年 Web 应用服务器市场可以说是 Nginx 一个人的舞台，凭借良好的性能表现、稳定可靠的发挥，Nginx 从 Lighttpd 等高性能 Web 应用服务器竞争者中脱颖而出，一路高歌猛进，短短几年时间就拿下了近百分之二十的市场份额。但是凭着历史积累下的庞大用户数，Apache 目前仍然是 最广为使用的 Web 服务器（根据 NetCraft 2015 年 2 月数据，在全球访问量最大的一百万个网站中 Apache 市场占有率为 50%，二三名分别为 Nginx 21%、IIS 12%），依然有大量历史长期项目以及更注重 Apache 丰富的功能且对性能要求不那么高的新项目选择并使用 Apache。

因为上述原因，在目前 Nginx 已逐步是多数互联网公司首选的情况下，我们仍包括了 Apache 的内容。

Apache HTTP Server（简称 Apache）

Apache 是一个跨平台、采用模块化设计的 Web 服务器，由于其简单高效、稳定安全的特性，被广泛应用于计算机技术的各个领域。

Apache 命令

```
httpd                        # 启动 Apache
httpd -h                     # 显示帮助
httpd -v                     # 显示版本信息
httpd -V                     # 显示版本，编译器版本和配置参数
httpd -t                     # 检查配置文件语法错误
httpd -l                     # 显示编译模块
httpd -L                     # 显示配置指令说明
httpd -f </path/to/config>   # 指定配置文件
httpd -S                     # 显示配置文件中的设定
```

一些 Apache 配置、模块

> 粗体部分 - 个人认为可多关注

- Options
  - 符号连接 - FollowSymLinks
    开启后，服务器将允许在此目录中使用符号连接，Apache 会检查每个请求中是否包含对符号链接的引用，对请求中包含的每个路径做一次 lstat() 系统调用。如考虑安全防护：永远不要允许使用符号链接；如考虑性能：永远使用 Options FollowSysLinks 且绝不使用 Options SysLinkIfOwnerMatch

    - 目录浏览 - Indexes, 建议关闭
- ServerRoot, DocumentRoot
- 网站默认首页文件 - DirectoryIndex
- 虚拟主机 **- NameVirtualHost, ServerName, VirutalHost, ServerAlias**
- **.htaccess** 支持 **- AllowOverride, .htaccess**
  只在必要目录中启用 AllowOverride（2.4 版本默认关闭）。.htaccess 文件可以极大便利 Apache 参数设置，而无需每次修改都要编辑 Apache 的配置文件（需要重启服务生效），但是也降低了服务器的性能。
- 工作模式/多处理模块（**MPM**）：
    - 查看当前工作模式：`httpd -l`
    - Prefork 模式
        - StartServers: 服务器启动时默认开启的进程数
        - MinSpareServers: 最小的空闲进程数
        - MaxSpareServers: 最大的空闲进程数
        - ServerLimit: 在Apache的生命周期内，限制MaxClients的最大值
        - MaxClients: 最大的并发请求数，最大值不能超过 ServerLimit 设置的值
        - MaxRequestsPerChild: 一个进程可以处理的最多的请求数（进程复用），如请求超过该设置则杀死进程，0表示永不过期。
    - Worker 模式
        - StartServers: 服务器启动时默认开启的进程数
        - MaxClients: 最大的并发请求数
        - MinSpareThreads: 最小的线程空闲数
        - MaxSpareThreads: 最大的线程空闲数
        - ThreadsPerChild: 每一个进程可以产生的线程数
        - MaxRequestsPerChild: 一个线程可以处理的最多的请求数（线程复用），如请求超过该设置则杀死线程，0表示永不过期。
    - Event 模式
- 日志级别、路径、格式 - LogLevel, ErrorLog, CustomLog
- 认证，授权与访问控制
    - **Order, Deny, Allow**, Apache: Access Control
    - Apache: Authentication and Authorization
- SSL 支持 - mod_ssl
- 避免 DNS 查询 - HostnameLookups off
- 请求超时 - Timeout
- 文件包含 - Include
- 缓存 - Cache Guide
- 代理 - Proxy, mod_proxy

扩展阅读

- **Apache 中文文档**
- **Apache: Frequently Asked Questions** - 英文，官方问题汇总，篇幅有点长，建议过一遍
- **Book: Apache Cookbook 中文版** - 关于 Apache 服务器安装、配置以及日常工作中可能遇到问题的解决办法的参考书
- **Developerworks: 将 Apache httpd 作为应用开发平台** - Apache 扩展开发快速入门示例

Nginx

Nginx ("engine x") 是一个高性能的 HTTP 和反向代理服务器，也是一个 IMAP/POP3/SMTP 邮件代理服务器 。 Nginx 是由 Igor Sysoev 为俄罗斯访问量第二的 Rambler.ru 站点开发的，自发布以来的很长一段时间里，Nginx 为 Yandex, Mail.Ru, VK, 以及 Rambler 等俄罗斯网站经受住了强大的访问考验。Igor 将源代码以类 BSD 许可证的形式（2-clause BSD-like license）发布。自 Nginx 发布以来，Nginx 已经因为它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而越来越受到各大互联网公司的欢迎。目前国内各大网站都已经部署了 Nginx，如新浪、网易、腾讯等，Nginx 技术在国内日趋火热，越来越多的网站开始部署 Nginx。

-- 源自：http://wiki.nginx.org/NginxChs

Nginx 命令

```
nginx                       # 启动 Nginx
nginx -h                    # 显示帮助
nginx -c </path/to/config>  # 指定一个 Nginx 配置文件，以代替缺省的。
nginx -t                    # 不运行，仅仅测试配置文件语法的正确性。
nginx -v                    # 显示 Nginx 的版本。
nginx -V                    # 显示 Nginx 的版本，编译器版本和配置参数。
nginx -s reload             # 更改了配置后无需重启 Nginx，平滑重启。
nginx -s stop               # 停止 Nginx
```

Nginx 常见使用场景/功能

- Web 服务器 - 提供虚拟主机服务、地址重写、默认错误页面等
- 反向代理 - 反向代理 Apache 服务、PHP-FPM 服务、Memecached 服务等
- 负载均衡 - 将请求根据设定的策略算法分发到一组服务器中的其中一台
- Web 内容静态缓存 - 缓存后端代理服务器提供的静态或动态内容
- 访问控制 - 通过 auth_basic, limit_req_zone, limit_req, limit_conn, limit_rate 等配置来限制用户访问、访问 IP、并发连接数、带宽使用情况等。

Nginx 模块

Nginx 的模块从功能角度主要可以分为以下三类，

- **Handler** 模块 - 主要负责处理客户端请求并产生待响应内容，比如 ngx_http_static_module 模块，负责客户端的静态页面请求处理并将对应的磁盘文件准备为响应内容输出。
- **Filter** 模块） - 主要负责对输出的内容进行处理，可以对输出进行修改，如 ngx_http_not_modified_filter_module, ngx_http_header_filter_module 模块。
- **Upstream** 模块 - 实现反向代理的功能，将真正的请求转发到后端服务器上，如 ngx_http_proxy_module、ngx_http_fastcgi_module 模块。

> Nginx 模块列表： http://wiki.nginx.org/Modules

Nginx 配置指令

Nginx 配置项指令（Directives）根据作用域/指令上下文（context）可分类如下，

- **main** - Nginx 在运行时与具体业务功能（比如 HTTP 服务或者 Email 服务代理）无关的一些参数，比如工作进程数，运行的用户等。
- **event** - 定义 Nginx 事件工作模式与连接数上限等参数。
- **http** - 与提供 HTTP 服务相关的一些配置参数。如是否使用 keepalive、是否使用gzip进行压缩等。
- **server** - HTTP 服务上支持若干虚拟主机。每个虚拟主机一个对应的 server 配置项，配置项里面包含该虚拟主机相关的配置。在提供 mail 服务的代理时，也可以建立若干 server，每个 server 通过监听的地址来区分。
- **location** - http 服务中，某些特定的 URL 对应的一系列配置项。
- **mail** - 实现 email 相关的 SMTP/IMAP/POP3 代理时，共享的一些配置项（因为可能实现多个代理，工作在多个监听地址上）。

> - Nginx 配置文件说明： 官方示例-英文，中文详解
> - Nginx 配置指令列表： 英文，中文

扩展阅读

- **Nginx** 性能调优 **-** 英文
- **Tengine** - Tengine 是由淘宝网基于 Nginx 发起的 Web 服务器项目（完美兼容 Nginx）。它在 Nginx 的基础上，针对大访问量网站的需求，添加了很多高级功能和特性。Tengine 的性能和稳定性已经在大型的网站如淘宝网，天猫商城等得到了很好的检验。
- **OpenResty** - OpenResty 通过汇聚各种 Nginx 模块，从而将 Nginx 有效的变成一个强大的 Web 应用服务器。这样，Web 开发人员可以使用 Lua 脚本语言调动 Nginx 支持的各种模块，快速构造出足以胜任 10K+ 并发连接响应的超高性能 Web 应用系统。
- **Nginx** 模块开发 **&** 原理剖析：**Nginx**开发从入门到精通 - 淘宝核心系统服务器平台组同

学组织编写的一本关于 Nginx 模块的开发以及它的内部原理的在线书。

Web 服务器常用配置

- Module comparison matrix - Nginx, Apache, Lighttpd 模块对应表
- 最小权限原则 设置文件、文件夹权限
  - chmod, chown
  - rwx? 755？ - Linux 文件系统权限 - 英文
- 单独建立运行用户及用户组
  - Apache - User, Group
  - Nginx - user
- 合理使用浏览器缓存 - Expires, Etag, Cache-Control, Last-Modified 浏览器缓存（Browser Caching）是为了节约网络的资源、加快浏览速度，浏览器在用户磁盘上对最近请求过的文档进行存储，当访问者再次请求这个页面时，浏览器尝试直接从本地磁盘获取文档内容，加速页面访问速度的一种方法。

  浏览器缓存主要有两种方式，

  - 缓存协商 - Etag, Last-modifie 缓存协商的方式去服务器端询问页面有没有修改过，没有修改则返回 304 直接使用缓存内容，否则返回新内容
  - 彻底缓存 - Cache-Control，Expires 彻底缓存的方式在缓存失效之前不再跟服务器交互，直接使用缓存内容

  您可以参考这篇文章了解到更多信息 - Heroku: Increasing Application Performance with HTTP Cache Headers

- 使用压缩 - 节省网络带宽
  - Apache - mod_deflate,
  - Nginx - gzip
- 保持连接 - KeepAlive 参数让服务器和客户端之间在指定一段时间保持同一个连接。这个特性有好处也有坏处，好处是如果客户端发出多个请求，服务端不必每次都花时间去创建连接，坏处是这段时间内即使客户端不再发出新的请求、访问新的页面，这个连接也会被占用，这对服务器资源来说是一种浪费。
  - Apache - KeepAlive, KeepAliveTimeout, MaxKeepAliveRequests
  - Nginx - keepalive, keepalive_requests, keepalive_timeout
- 配置文件推荐目录结构

```
mkdir ./site-available ./site-enabled
echo 在 site-available 中写好网站配置文件
cd ../site-enabled
ln -s ../site-available/xxx.conf
```

- 计算进程平均占用内存 (kb) - 以便决定服务器需要多少内存空间（以 Apache 为例）

  ```
  ps aux | grep -v grep | awk '/sbin\/httpd/ {sum+=$6;n++}; END {print sum/n}'
  ```

- 地址重写 - URL Rewrite

  > Tool：Apache 规则转 Nginx 规则

  - Apache - mod_rewirte 模块
  - Nginx - ngx_http_rewrite_module 模块
- 响应头不显示版本信息及操作系统版本信息
  - Apache
    - ServerTokens - 是否在错误页面、目录列表等页面的底部显示服务器操作系统信息
    - ServerSignature - HTTP 头中是否回显 Apache 版本信息
  - Nginx - server_tokens off;
- 关闭不用的模块及功能
  - Apache - 注释掉不用的 LoadModule 配置项
  - Nginx - 编译时确定（Tengine 支持运行时动态加载 HTTP 模块）

扩展阅读

- Tutorial: Top 20 Nginx WebServer Best Security Practices
- Book: 构建高性能Web站点
- Book: 高性能网站建设指南
- Book: 高性能网站建设进阶指南

数据库服务器

MySQL

MySQL 分支与变种

- **MySQL**
- **MariaDB**
- **Percona Server**

MySQL 知识点

- SQL 基础
  - DDL(Data Definition Language) - 数据定义
    - CREATE - 创建表
    - ALTER - 修改表
    - DROP - 删除表
  - DML(Data Manipulation Language) - 数据操作
    - INSERT - 数据插入
    - DELETE - 数据删除
    - UPDATE - 数据修改
    - SELECT - 数据查询
      - GROUP - 分组
      - JOIN - 连接
      - UNION - 联合
      - 子查询
      - 分页 - LIMIT, OFFSET
  - DCL(Data Control Language) - 数据控制
    - GRANT - 授权
    - REVOKE - 撤销授权
  - 事务(Transactions)
    - ACID 特性
    - COMMIT
    - ROLLBACK
  - 数据提交方式
    - 显式提交 - COMMIT 命令直接完成的提交为显式提交
    - 隐式提交 - SQL 命令间接完成的提交为隐式提交

        - 自动提交 - AUTOCOMMIT ON
    - 范式 & 反范式
- 存储引擎
    - MyISAM
    - InnoDB
- 权限
    - 最小权限原则
    - Grant
    - Revoke
    - 系统表 user, db, host, tables_priv, columns_priv, procs_priv
- 字符集与校对
- 索引
- 约束
    - 外键约束
- 视图
- 查询优化
    - Explain
    - Index Hint
    - 慢查询
- 服务器状态
- 主从分离 & 复制
- 分区表

MySQL 配置

相关工具

- PHPMyAdmin
- HeidiSQL
- Navicat（收费）
- Sequel Pro - Mac OS
- MySQL Workbench

扩展阅读

- Book: 高性能 MySQL
- Blog: MySQL Performance Blog

MongoDB

相关工具

PHP

各版本主要功能改动

- PHP 5.3 - namespace, closure
- PHP 5.4 - trait, short array syntax
  - http://php.net/manual/en/migration54.new-features.php
- PHP 5.5 - finally, generator
  - http://php.net/manual/en/migration55.new-features.php
- PHP 5.6 - constant expressions, variadic function, argument unpacking, phpdbg
  - http://php.net/manual/en/migration56.new-features.php
- PHP 7.0 - Improved **performance**: PHP 7 is up to twice as fast as PHP 5.6, Consistent 64-bit support, Many fatal errors are now Exceptions, Removal of old and unsupported SAPIs and extensions, The null coalescing operator (??), Combined comparison Operator (<=>), Return Type Declarations, Scalar Type Declarations, Anonymous Classes
  - http://php.net/manual/en/migration70.new-features.php

PHP 命令

PHP 知识结构

> - PHP 语言基础请参考 PHP 官方文档
> - 关于安全、设计模式等更多内容请参考「设计&开发」相关章节

- PHP 生命周期
- PHP FPM (FastCGI Process Manager)
- 命令行模式
- 语言结构(Language Construct) - language construct is not a function, cannot be called using variable functions, such as echo, print, unset(), isset(), empty(), include, require
  - Stack Overflow: Language Construct vs Function
- 数组操作（个人认为 PHP 与其他语言对比，强大的数组功能大大简化了开发者对数据的结构类组织工作）
  - 关联数组
  - 多维数组
  - 排序

- 自动加载 - Autoloading
    - __autoload
    - spl_autoload_register()
- 类型比较 - PHP type comparison tables
    - isset(), empty(), is_null(), boolean : if($x) 函数对变量 $x 进行比较
    - 松散比较 == vs 严格比较 ===
- 魔术方法 - magic method

> PHP 将所有以 __（两个下划线）开头的类方法保留为魔术方法。所以在定义类方法时，除了上述魔术方法，建议不要以 __ 为前缀。

- 属性重载
    - __get() - 调用未定义的属性时调用
    - __set() - 对未定义的属性赋值时调用
    - __isset() - 对未定义属性使用 isset()、empty() 方法时被调用
    - __unset() - 对未定义属性使用 unset() 方法时被调用
- 方法重载
    - __call() - 调用一个不存在的方法时被调用
    - __callStatic() - 调用一个不存在的静态方法时被调用
- 序列化
    - __sleep() - 序列化对象时被调用
    - __wakeup() - 反序列化对象时被调用
- 构造函数
    - __construct() - 构造函数，对象初始化时被调用
    - __destruct() - 析构函数，对象释放时被调用
- 其他
    - __invoke() - 尝试以方法调用的方式调用一个对象时被调用
    - __toString() - 当一个类被尝试当做字符串使用时被调用，返回值只能为字符串，否则触发 PHP 致命错误
    - __clone() - 使用 clone 关键字作对象复制时被调用
    - __set_state() - 使用 var_export() 方法导出类时被调用
    - __debugInfo() - 使用 var_dump() 方法导出类时被调用

- OOP - 面向对象
    - 继承、封装、多态
    - SPL - Standard PHP Library
    - 抽象类 vs 接口
    - 访问控制 - public, private, protected
    - 属性
    - 静态方法、属性
    - 静态延时绑定 - static
    - 常量 vs define

  - 命名空间
  - 反射
  - 类函数 vs 对象函数
  - Final 类
  - Traits - like Partial Class in C#
- OPCode
- 回调、匿名函数和闭包
- 生成器 - Generator
  - yield
- heredoc & nowdoc (PHP 5.3.0 新增)
- PDO & mysqlnd
- 错误和异常处理
  - PHP Error
    - trigger_error
    - 将 PHP Error 转为 Exception 处理 - set_error_handler
  - PHP Exception
- Coding Tips - 语言层面编写高性能 PHP 代码 - 单引号比双引号快？ echo 比 print 好？

  > NOTE: 两篇文章都有一段时间没有更新了，不过还是有一定的参考价值

  - Collection of PHP Performance Benchmarks
  - PHP Benchmark
- 安全模式 - safe_mode（自 PHP 5.3.0 起废弃并将自 PHP 5.4.0 起移除。）

扩展阅读

- **Book: PHP 和 MySQL Web 开发 - PHP 5.3/2009** - PHP 开发入门最佳选择，主要关注语言本身
- **Book: 深入PHP 面向对象、模式与实践** - PHP 开发进阶，关注使用面向对象开发、设计模式编写稳定可维护的代码，同时介绍了版本管理(SVN)、持续集成等实践知识
- **Book: Morden PHP - PHP 5.6/2015** - PHP: The Right Way 作者新作，关于 PHP 开发实践的一本书
- **Github: Awesome PHP** - PHP 相关库、资源整理
- **Github: PHP must watch** - PHP 相关视频整理

网站系统架构

一个成熟的大型网站（如百度、淘宝、京东等）的系统架构并不是开始设计就具备完整的高性能、高可用、可扩展、安全等特性，它总是随着用户量的增加、业务功能的扩展逐渐演变完善的，在这个过程中，开发模式、技术架构、设计思想也会发生很大的变化，技术团队也从几个人发展到一个部门甚至产品线。成熟的系统架构是由小及大、从无到有，随着业务发展渐进式完善、发展出来的，并不是一开始就全部开发好了的。

下面将简要介绍广泛运行在大型网站系统架构中一些常见的技术和手段。

内容缓存加速

软件名称	性能	功能	过滤规则配置
Squid	不能多核是硬伤，磁盘缓存容量有优势，性能中等	多，支持ACL角色控制，也支持ICP缓存协议	支持外部规则文件读取及热加载，支持热启动
Varnish	多核支持，内存缓存，性能强	够用，不支持集群，支持后端存活检查	不支持外部文件读取，需要转义，支持热启动
Nginx	多核支持，支持代理插件，性能较强	多，通过插件可以充当多角色服务器	不支持外部文件读取，需要转义，支持热启动
ATS	多核支持，磁盘/内存缓存，性能强	够用，支持插件开发，也支持ICP协议	支持外部规则文件读取及热加载，支持热启动，但缺乏文档
HAProxy	多核支持，无缓存，HTTP头支持语法操作，性能强	少，只专注HTTP头部解析和转发功能，支持ACL角色控制，支持后端存活检查	支持外部规则文件读取及热加载，支持热启动，支持会话粘滞和长连接

来源：又拍云存储 - CDN 架构探索 by 邵海洋 @2014

- **Squid** - 功能全而大，磁盘缓存，适合于各种静态的文件缓存（截止至目前为止相对使用最为广泛）
- **Varnish** - 内存缓存（少数人的玩具），性能强，对小文件如 CSS, JavaScript, 小图片之类的支持很棒
- **ATS - Apache Traffic Server** - 磁盘/内存缓存，性能强，支持插件开发
- **Nginx** - 代理功能只是 Nginx 的一个模块功能，得益于 Nginx 强大的性能，目前适合缓存少量页面资源

服务负载均衡

> 现在对网络负载均衡的使用是随着网站规模的提升根据不同的阶段来使用不同的技术：
>
> 第一阶段：利用 Nginx 或 HAProxy 进行单点的负载均衡，这一阶段服务器规模刚脱离开单服务器、单数据库的模式，需要一定的负载均衡，但是仍然规模较小没有专业的维护团队来进行维护，也没有需要进行大规模的网站部署。这样利用 Nginx 或 HAproxy 就是第一选择，此时这些东西上手快， 配置容易，在七层之上利用 HTTP 协议就可以。这时是第一选择。
>
> 第二阶段：随着网络服务进一步扩大，这时单点的 Nginx 已经不能满足，这时使用 LVS 或者商用 Array 就是首要选择，Nginx 此时就作为 LVS 或者 Array 的节点来使用，具体 LVS 或 Array 的是选择是根据公司规模和预算来选择，Array 的应用交付功能非常强大，本人在某项目中使用过，性价比也远高于 F5，商用首选！但是一般来说这阶段相关人才跟不上业务的提升，所以购买商业负载均衡已经成为了必经之路。
>
> 第三阶段：这时网络服务已经成为主流产品，此时随着公司知名度也进一步扩展，相关人才的能力以及数量也随之提升，这时无论从开发适合自身产品的定制，以及降低成本来讲开源的 LVS，已经成为首选，这时 LVS 会成为主流。 最终形成比较理想的基本架构为：Array/LVS —> Nginx/Haproxy —> Squid/Varnish —> AppServer。
>
> 来源：**Nginx/LVS/HAProxy 负载均衡软件的优缺点详解 by 博客教主 @**

- **Keepalived** - Keepalived 主要用来防止单点故障（单点故障是指一旦某一点出现故障就会导致整个系统架构不可用）的发生

  keepalived 是基于 VRRP 协议（VRRP 协议介绍）的，请一定先了解 VRRP 协议后再进行配置。keepalived 可以把多台设备虚拟出一个 IP，并自动在故障节点与备用节点之间实现 failover 切换。这样我们配置两台货多台lvs调度节点，然后配置好 keepalived 就可以做到 lvs 调度节点出现故障后，自动切换到备用调度节点。（同样适用于 MySQL，Nginx 等）

  - 《keepalived 权威指南》
- **Nginx**
- **HAProxy**
- **LVS(Linux Virtual Server)** - 淘宝正明（章文嵩大大）在国防科技大学读博期间的作品，是国内最早的开源软件之一

分布式缓存系统

- **Memcached**
- **Redis**
  - **SSDB** - A fast NoSQL database, an alternative to Redis

分布式文件系统

- FastDFS
- MogileFS
- HDFS

MySQL 数据库集群

主从复制，读写分离，分库分表

- Amoeba
- MySQL Proxy

NoSQL 数据库

- Cassandra
- MongoDB
- Hadoop & HBase
- Riak
- CouchDB
- Neo4j

分布式消息队列

- RabbitMQ
- Kafka
- ActiveMQ
- ZeroMQ
- Kestrel
- Gearman

扩展阅读

- Queues.io - 该项目汇集整理了目前流行的各种消息队列/任务队列系统

搜索引擎技术

- Lucene 及其变种

    - **Elasticsearch**
    - **Solr**
- **Sphinx Search**
- **Xapian**

扩展阅读

- Elasticsearch vs. Sphinx Comparison

网站/服务运行监控

服务器硬件/系统监控

- Nagios
- Cacti
- Zabbix
- Icinga
- Munin
- Ganglia
- sensu

日志收集、处理、可视化

- logstash - aka. ELK (Elasticsearch, Logstash, Kibana)
- Splunk - 商业版，后来者 ELK 的主要假想超越对象
- Sentry
- loggly
- papertrail
- Sumo Logic

应用/服务可用性监控

- APM - Application Performance Monitoring
    - New Relic
    - 听云 - 基调网络
    - OneAPM
- 服务或应用当前状态 - Status page for your app or website
    - StatusPage
    - Cachet - PHP 开源版本
    - Stashboard - Python 开源版本

- App Dynamics
- ruxit
- Datadog
- Keynote
- Pingdom
- Uptime Robot
- Monitority
- 小蜜蜂
- 监控宝

分布式服务

扩展阅读

- Book: 大型网站技术架构:核心原理与案例分析

一致性开发环境

「工欲善其事，必先利其器!」

然后，比「 XXX 是最好的语言」更难的问题来了！ XXX 是最好的操作系统、编辑器、IDE（集成编辑环境）。。。连 Wikipedia 上都创建了专门的页面。。。 :-X

- Operating system advocacy
- Editor war

> 「文无第一，武无第二」，个人无意更无力解决「程序员鄙视链」难题， 能输出才是王道，我们更应该「放弃编程技术好坏之争，着眼于解决问题」 -- 知乎@陈能干

保证程序员本地开发环境与服务器运行时环境一致，可以提高大大降低代码上线后由于运行环境不一致导致的兼容性问题，减少不必要的沟通和学习成本，提高程序员开发效率。

虚拟化技术

在虚拟机流行以前，为了确保多位程序员开发出的代码能在服务器环境上正确运行，很多公司的办法是组里提供独立的开发服务器，对团队成员的开发运行环境没有统一的模板或标准，由各成员自行配置专属的开发环境，开发过程中/上线前将代码提交到服务器验证代码是否正确。

VMWare Workstation/Virutal Box

- Share Folder
- Unity Mode
- Snapshot

Vagrant

Docker

集成环境安装包

很多初次接触 LAMP 开发的同学对手动配置开发环境感到麻烦，因为这需要将 PHP、Apache/Nginx/Lighthttpd、MySQL/MongoDB 等应用软件包分别下载下来然后分别进行安装配置。AMP 集成环境安装包的目标正是为开发人员提供这方面的便利，它们可以一次安装配置好 Apache、MySQL、PHP，甚至将常用的数据库管理软件 phpMyAdmin 等一并安装好，以便大家装完就能投入使用...

目前常见的 AMP 集成环境安装包有 XAMPP、WAMPServer、AMPPS 等，

- **Wikipedia: List of Apache–MySQL–PHP packages**

> 类似一体化的 AMP 集成环境安装包太多了，虽然确实会省点事，但个人不推荐作为日常使用，还是建议使用集成环境的同学从零开始对手动安装配置环境详尽仔细的研究清楚，毕竟这是开发人员必不可缺的基础技能。

编辑器

- IDE vs Text Editor
- Eclipse PDT (Eclipse PHP Development Tools)
- PHPStorm
- Sublime Text
- Vim
- Emacs
- ...

扩展阅读

- **PHPStorm 短视频系列教程：Be Awesome in PHPStorm** - 英文，推荐
- **JetBrains 官方短视频系列教程：PhpStorm Video Tutorial** - 英文，不过比利时兄弟的口音真心让人醉了。。。

设计 & 开发

开发规范

代码规范

PEAR Standard

FIG (Framework Interop Group) Standards

- **PSR-0 (Autoloading Standard) (Deprecated, use PSR-4 instead)**

  > 感谢 @lifesign 同学指出。FIG 在 2014-10-21 宣布 PSR-4 为类自动加载的推荐标准。如果您的项目不再考虑支持 PHP 5.2 及以下版本，那么请直接接使用或升级支持 PSR-4 标准。

  PSR-0，关于 类自动加载。因为 PHP 类加载的机制本质是通过 include 方式达到的，为了避免大量手动 include，需要通过良好的代码组织规范实现自动加载。

  PSR-0 通过时，PHP 5.3 稳定版本才发布没多久，社区内绝大部分开发者支持的仍然为 PHP 5.2 及以下版本（没有 namespace 特性，普遍使用下划线做类隔离），主流开源项目（以 Zend 为盛）也大多遵从 PEAR 标准或自定标准，不过，伴随着 PHP 5.3 的普及（写这段文字的时候 PHP 5.3 已经停止官方支持了）以及 Composer 社区的蓬勃发展，PSR-0 逐步为各大开源项目所接受，取代 PEAR 成为 PHP 开源社区非官方标准（鼓掌）。话说回来，毕竟这是 FIG 组织 2009 年成立推出的第一个规范，当时又大量需要考虑对主流项目的支持，这或许导致了 PSR-0 的一些问题，而这正是自动加载标准修正案（PSR-4）推出的原因。

  > 扩展阅读：
  >
  > - PSR-0 Standard
  > - spl_autoload_register()
  > - Send PSR-0 to the Standards Farm in the Sky

- PSR-1 (Basic Coding Standard)
- PSR-2 (Coding Style Guide)
- PSR-3 (Logger Interface)
- PSR-4 (Improved Autoloading)

PHP Code Sniffer

包管理 (Component Management)

- PEAR
- Composer
- PECL
- Pickle

版本命名规范

- Sematic Versioning
- Software release life cycle

程序设计编码

- 设计模式 (Design pattern)
- 设计语言 (Design Language)
    - UML
- SOA
    - Restful API
    - RPC
- 代码重构 (Code Refactor)
- 代码评审 (Code Review)

测试

- 单元测试 (Unit Testing)
  - PHPUnit
- 代码覆盖率 (Code Coverage)

诊断与调优

- 代码调试 (Debugging)
  - var_dump()
  - PHP单步调试
    - Xdebug
    - Zend Debugger
  - DebugToolbar
  - GDB
    - 用GDB调试程序
    - Debugging PHP segfault backtraces with gdb
- 网络诊断 (Network)
  - Fiddler
    - LivePool by Tencent AlloyTeam
  - weinre (WEb INspector REmote)
  - Browser Inspector
    - Firebug - 抓包与各种调试
    - Tamper Data - 拦截修改
    - Live Http Header - 重放功能
    - Hackbar - 编码解码/POST提交
    - Modify Headers - 修改头部
  - Wireshark
- 代码分析 (Code Analysis)
  - XHProf & XHGui
  - Xdebug profile & valgrind
  - PHPMD - PHP Mess Detector
  - phptrace
- 日志记录与处理 (Logging)
  - Monolog
  - Splunk
  - ELK (ElasticSearch, Logstash, Kibana)

安全 (Security)

常见漏洞

- XSS(Cross-site scripting)
- SQL Injection
- Command Injection
- Code Execution
- File Disclosure
- File Inclusion
- CSRF(Cross-site request forgery)
- Path Traversal
- Code injection

相关资源

- OWASP (Open Web Application Security Project)
- Survive The Deep End: PHP Security
- CVE List
- RIPS 静态代码分析工具
- Freebuf
- 乌云

文档

如何为项目写文档、用什么写文档、如何方便的管理/更新文档，这是很多技术人员都要经常面对的一些问题。根据个人经验，程序开发类工作中我们涉及到的文档可能主要包括下面几类，

- 函数/类的接口说明文档， 比如 XXX Framework API Documentation
- 服务端接口说明文档，比如 Web Service API, Restful API 等
- 使用指南、手册等说明文档，比如 Getting Start XXX, XXX Tutorial, XXX Guideline, XXX Manual 等
- 设计相关文档，比如 数据库 ER 图、类图、时序图等（Enterprise Architect）

相关工具/平台

下面，我们将分别介绍一些常用的相关工具/平台

使用指南、手册等说明文档

此类文档内容多以大段文字说明为主，一般按章节进行内容组织，随着项目的不断成熟、演进，很多成熟项目的文档都会有相当篇幅；于此同时，出于格式、排版等需求，逐步形成了多种标记语言标准，常见的比如 TeX, DocBook, Markdown, reStructuredText, AsciiDoc 等。目前互联网项目中主要流行 Markdown, reStructuredText, AsciiDoc 等轻量级标记语言，Tex、DocBook 过于复杂，reStructuredText 主要在 Python 开发者中流行，Org-mode 主要以 Emacs 用户为主，Markdown 最为流行（由于 Markdown 过于简单，目前几大组织/公司分别定义了自己的 Markdown 扩展语法，比如 GitHub Flavored Markdown，MultiMarkdown，Markdown Extra，Pandoc Markdown 等）。

扩展阅读：

- Comparison of document markup languages
- Lightweight markup language - 其中有好几种都是 Markdown 的方言；毛主席教导我们要辩证的去看问题，多种方言同时出现一方面反映出以简为美起家的 Markdown 是如此流行，另一方面也显示出 Markdown 扩展语法标准混乱的无奈。
- Lightweight Markup: Markdown, reStructuredText, MediaWiki, AsciiDoc, Org-mode

ReST（reStructuredText）

- Sphinx Documentation，可能是 ReST（reStructuredText） 系的代表项目了吧，Python 开发，由于语法简单、功能强大，非常多的项目都使用 Sphinx 来生成项目文档（感受

下?），跟他一起出现的还有 Read The Docs（开源文档生成的托管平台和社区，同时支持 reStructuredText 格式以及 Markdown 格式） 这位兄弟，关于他们之间的关系，可以看下这里；相关 PHP 项目示例，

  - Guzzle，源码
  - CakePHP Cookbook 3.x

Markdown

由于 Markdown 简单易读，只需要普通的文本编辑器就能够编辑，语法更是简单到每个人都可以在数分钟内学会，因此，作为一门 2004 年才被发明出来的轻量级标记语言，Markdown 很快便得到了诸如 Github，reddit, Stack Exchange, Bitbucket 等大量网站的支持。而伴随着 Github 旋风横扫整个开源界， git + Github + Markdown + Jekyll 组合更是成为互联网时代大量科技写作者甚至小部分文艺青年的主力生产力平台。

- Jekyll - 不仅仅可以生成静态博客站点，Jekyll 同样适用于文档内容生成，Wista（一家企业视频托管服务提供商）在这里分享了他们基于 Jekyll 的文档管理经验，你可以点这里先睹为快（源码）。下面列举了 Jekyll 文档其他相关主题，
  - Jekyll Documentation Theme，示例：Official Doc
  - Jekyll-docs-template，示例：Official Doc，ModelTree
- MkDocs - 根据 Markdown 文档生成静态网站，Python 语言开发，
  - SlimFramework v2 Doc - 个人常用的一个 PHP 轻量级框架，小内核 + Composer 包，可玩性很高~ 类似的可选项还有 Silex 和 Lumen
  - 小米开放平台文档中心
  - Flask API Doc - 另外一款轻量级 Python Web 开发框架
- GitBook - 除了 Markdown，GitBook 还支持另外一种与 Markdown 很类似但功能更丰富的标记语言 AsciiDoc；GitBook 除了提供基于 node.js 开发的可供本地运行的命令行版本外，还提供了在线托管服务。
  - GitBook Multi-Languages support
- Peach - 国人基于 Go 语言开发的一款支持多语言、实时同步以及全文搜索功能的 Web 文档服务器。赞一个~，使用示例，
  - Peach 文档 - Peach 官方文档
  - Gogs 文档 - 国人基于 Go 语言开发的类 Gitlab 服务器，界面比较小清新哦~

下面的表格对比了几款支持 Markdown 的主流文档工具之间主要特性的区别（实际区别可能与我所理解的有偏差）：

名称	自助托管----	多语言文档	实时同步	静态 HTML	多版本
Read The Docs	✗	✓	✓	✓	✓
Peach	✓	✓	✓	✓（可缓存）	计划中
Mkdocs	✓	✗	✗	✓	✗
GitBook	✓	✓	✓	✓	✗
Jekyll	✓	✓（插件）	✓	✓	✓（不完美）

AsciiDoc

AsciiDoc 最初基于 Python 开发，其基本语法跟 Markdown 类似，例如以 = 开头作为标题、以 * 号开头作为列表项等，详细的语法说明可以看这里，但 AsciiDoc 比 Markdown 支持更多的格式，比如表格、文件包含等，同时 AsciiDoc 是 O'Reilly 的 Atlas 在线出版平台 的推荐语言，相当大部分的 Git 官方文档都使用了 AsciiDoc 格式。

扩展阅读：

- AsciiDoctor - 2013 年发布，是 AsciiDoc 基于 Ruby 语言的实现，主要在 GitHub 使用。

DocBook

- DocBook 比 HTML 语言还早一年（1992 年）出现的 XML 系标记语言，可以非常方便的生成其他文件格式，比如 HTMl、PDF、CHM 等，开源界使用 DocBook 的项目非常多，比如大家耳熟能详的一些项目，Linux Kernel、FreeBSD、PostgreSQL、O'Reilly Media、OpenStack，PHP 的官方文档也用的它哦，更多项目和组织列表可以看这里；不过纯 PHP 相关项目使用 Docbook 的较少，毕竟使用 XML 来写文档门槛有点高 :sunglasses:，项目示例，
    - PHPUnit Documentation，源码

Tex

Tex 是由 Donald Knuth 创造的基于低级编程语言的电子排版系统，利用 TeX 能够对文章进行十分精美的排版，TeX 提供了一套功能强大并且十分灵活的排版语言，同时 TeX 系统是公认的数学公式排得最好的系统。

- LaTeX - LaTeX（发音为“Lah-tech”或“Lay-tech”）是由 Leslie Lamport 开发的当今世界上最流行和使用最为广泛的TeX宏集。 把 LaTeX 放到最后是由于 TeX 语法的高门槛，导致除了专业的科学研究、出版领域，除了我们敬仰的那些大神们，在偏工程/应用类的普通

码农界中没有真正大规模流行开来。

函数/类的接口说明文档

- PHPDoc，写 PHP 的都应该知道这个哈，嘿嘿。示例，
  - Zend Framework 2.4 API Documentation
  - Offical Demo with Clean2 theme
- Sami，Symfony 框架使用的文档生成工具，示例，
  - Symfony 3.0 API
  - Composer API
- Doxygen，超级老牌文档生成工具了，除了 PHP 外还支持其他 N 种编程语言
- phpDox
- Apigen
  - Amazon AWS SDK for PHP 3.x
  - CakePHP API Docs

服务端接口说明文档

- Swagger, Demo
- Apiary - （推荐）Apiary.io平台具有协同设计、即时API模拟、快速生成源码、自动测试和代码调试的开源设计工具，最重要的是可以在线模拟测试，因为该平台具备模拟服务器测试服务，可以把设计好的程序在线测试、验证。
- Apidoc, Demo
- Slate - 创建好看的 API 文档，Travis-CI, Appium 等项目使用中，更多例子：Examples of Slate in the Wild
- JSON-Schema
- prmd - JSON Schema tools and doc generation for HTTP APIs
- 如何编写 API 的文档？

设计相关文档

过程 & 实践

过程管理/生命周期

- 瀑布式开发
    - PCM
    - RUP
- 敏捷开发
    - Agile & Scrum
        - 硝烟中的Scrum和XP(http://www.infoq.com/cn/minibooks/scrum-xp-from-the-trenches)
    - Kanban

版本管理

> 眼下最流行的"版本管理系统"，非 Git 莫属！
>
> 关于是否使用 SVN？个人的建议是如果团队有权限管理、代码安全控制等考虑的话，那么考虑 SVN，否则建议使用 Git/Mercurial，
> 关于 Mercurial 还是 Git，因为 GitHub 这个大社区的缘故，因此如果是程序员个人搞开源的话，还是建议优先 Git，如果 Git、Mercurial 您都想试试，那么这里打个广告：强烈建议您试试 BitBucket，Mercurial 和 Git 都支持，重要的是支持无限个私有库（业界良心），用来搞私人项目真心好，1024 个赞！Google Code 在 2008 年曾经有一个关于 Git or Mercurial 的分析文章（DVCSAnalysis : Analysis of Git and Mercurial），不过时间很久了，仅供参考。Google Code 也即将被大扫除掉了~ :-(

Git

- Git 简明教程 - （推荐）短小精悍纯干货！如果您是初次接触 Git 想要快速入门，那么强烈推荐您先看这个
- Git 提交的正确姿势：Commit message 和 Change log 编写指南 - （推荐）阮一峰老师最新出品，虽然讲的 Git，个人认为同样适用其他版本管理系统

Git Flow

- Blog: A successful Git branching model - 基本上是目前为止最靠谱的 Git Flow 指南了，也可以参考阮一峰老师的解读 - Blog: Git 分支管理策略
- Atlassian Git Tutorials - Atlassian 关于 Git 的教程（推荐）。Comparing Workflows 这个章节介绍了几种常见的 Git 协作形式，Comparing Workflows 中文翻译：Git 工作流指南
- LeanCloud 团队的 Git 分支管理规范 - LeanCloud 的这个开放资源项目逼格很高，里面很多干货，请收下我的膝盖 (逃~

Github Flow

- Tutorial: Understanding the GitHub Flow - Github 官方指南
- Blog: Github flow - Scott Chacon 在 Github 工作，更是《Pro Git Book》作者

扩展阅读

- Book: Pro Git Book - 关于 Git 这一本书够了，入门先看前面几章了解基本概念和日常使

用
    - Pro Git 第一版 V1
    - Pro Git 第二版 V2
- GitHub Cheatsheet - 这个项目整理了许多关于 Github & Git 的小技巧，强烈建议过一下
- GotGitHub - Git 权威指南作者蒋鑫关于 Github 写的开源书
- Git Rebase - 洁癖者用 Git：pull --rebase 和 merge --no-ff

Subversion/SVN

SVN 版本库布局 - trunk, branches, tags

SVN Flow? :-)

SVN 版本库只是一个包含目录和文件的文件系统，而且它的文件系统是版本化的，并且实现了"廉价"拷贝，让它的这种操作比传统文件系统廉价很多，但是版本库本身还是像一个文件系统。

至于版本库中目录的名称，只是一种习惯，他们在 SVN 中没有特别含义。我们在一些著名开源项目的版本库中，通常可以看到 trunk、branches、tags 等三个目录。

由于 SVN 固有的特点，目录在SVN 中并没有特别的意义，但是这三个目录却在大多数开源项目中存在，这是因为这三个目录反映了软件开发的通常模式。

- trunk 是主分支，可以认为是项目的开发主线，你可以称之为 main、master 或任何你喜欢的名字。
- branches 是分支。一些阶段性的 release 版本，这些版本是可以继续进行开发和维护的，则放在 branches 目录中。人们因各种目的使用分支，你或许希望通过特性分支或客户修改分支来隔离你的发布或维护分支等。
- tags 目录一般是只读的，这里存储阶段性的发布版本，对作为里程碑的版本进行存档。通常情况下 tags 与分支的区别就是 tags 一旦创建不能修改，你也可以将标签目录叫做 releases 或任何你喜欢的名字

开源社区采纳了多种常用布局作为最佳实践，我们应该遵循推荐的布局方式，按照这些布局约定，大家可以方便的访问版本库找到所需要的信息。

有两种常见的版本库布局：

第一种布局是版本库包含一个项目或一组紧密联系项目的最佳选择。

```
Project
  ├─trunk
  ├─branches
  └─tags
```

这个布局非常好用，因为分支与标签整个项目或一组项目会非常简单，只需要一个简单的命令：

```
svn copy url://repos/trunk url://repos/tags/tagname -m “Create tagname”
```

这可能是最常用的版本库布局，被许多开源项目采用，就像 Subversion 本身和 Subclipse，这是大多数主机站点，如Tigris.org、SourceForge.net 和 Google Code 遵循的方法，这些站点的每个项目有自己的版本库。

另一种布局是针对一个版本库包含不相关项目的最佳选择。

```
ProjectA
  ├─trunk
  ├─branches
  └─tags
ProjectB
  ├─trunk
  ├─branches
  └─tags
```

在这种布局里，每个项目会存在顶级目录里，然后该目录之下创建 trunk/branches/tags，其中与第一种布局相同，这只是将项目放到自己版本库方式的替换，他们都在一个版本库中。Apache 软件基金会使用这种布局方式来存放他们的所有项目在一个版本库。

通过这种布局，每个项目都有自己的分支和标签，可以很容易使用一个命令创建分支和标签，就像前面展示的：

```
svn copy url://repos/ProjectA/trunk url://repos/ProjectA/tags/tagname -m “Create tagna
me”
```

这种布局可以简单的创建同时包含 ProjectA 和 ProjectB 的标签，你可以这样做，但是需要多个命令，你也要决定是否创建一个特别的目录存放这种分支和标签，如果你需要经常这样做，你或许应该考虑第一种布局。

扩展阅读

- Book: Subversion 与版本控制

Mercurial/hg

扩展阅读

- Book: Mercurial: The Definitive Guide
- Learning Mercurial in Workflows
- Blog: An Introduction to Hgflow
- Blog: A Guide to Branching in Mercurial

相关工具

服务端

- Gitlab - 自建 Git 平台的多数都用 ta 啦，Github fork~？
- Gogs - Go Git Service - Go 语言开发的轻量级 Git 管理工具（不喜欢随大流的同学们有福啦~）
- VisualSVN - Windows 平台下很多人的选择
- iF.SVNAdmin - Web 版 SVN 管理工具(PHP)
- USVN - Web 版 SVN 管理工具(PHP)

客户端

- 乌龟系列（Windows only）
    - TortoiseSVN - Subversion client for Windows
    - TortoiseGit - Git client for Windows
    - TortoiseHg - Mercurial client for Windows, Mac OS and Linux
- Eclipse 系列
    - EGit - Eclipse 中的 Git 插件
    - Subclipse - Ecplise 中的 Subversion 插件
    - Subversive - Eclipse 中的 Subversion 插件
- RabbitVCS - Linux 下很好用的一款图形客户端，支持 Subversion, Git & Mercurial
- GitHub Desktop - Github 提供的 Git 客户端，提供 Mac OS 和 Windows 版本
- SourceTree - Atlassian 提供的同时支持 Mac OS 和 Windows 系统的一款图形客户端，支持 Git & Mercurial
- SmartSVN - Subversion client for Mac OS, Windows and Linux（专业版收费）
- Cornerstone - Mac OS 下的 SVN 客户端（收费）
- Versions - Mac OS 下的 SVN 客户端（收费）
- Tower - Mac OS 平台 Git 客户端（收费）
- Wikipedia: Comparison of Subversion clients
- git-scm: More GUI client from git-scm.com

持续集成 (Continue Integration)

- Phing
- Jenkins CI
- Travis CI
- StyleCI

上线部署

- Puppet
- Chef
- Ansible

PHP Internals

PHP扩展开发

- PHP 源码
  - Source Insight
- 相关资料
  - Extending and Embedding PHP
  - PHP Internals Book
  - 深入理解PHP内核
  - PHP扩展开发及内核应用
  - Laruence 鸟哥博客

PHP Virtual Machine

- HHVM
- HippyVM